

The Bioinformatics Lab Challenge: Webserver

Markus Meier

June 27, 2011

Contents

0.1	Introduction	2
0.2	Default Configuration	2
0.3	Virtual Host	3
0.4	HyperText Transfer Protocol Secure	4
0.4.1	Certificates	5
0.4.2	Configuring Apache to use HTTPS	5
0.5	PHP	6
0.6	Access Restriction	6
0.6.1	Restriction by IP	6
0.6.2	Restriction by User/Password	7
0.7	LDAP-Administration	8
0.7.1	phpldapadmin	8
0.7.2	ldap-account-manager	8
0.8	References	9

0.1 Introduction

In this challenge we will set up a webserver - namely the Apache webserver. This webserver is developed since 1995. Since 1996 it has been the most popular HTTP server software. In an estimation of 2011 Apache is used for 63% of all and for 66% of the million busiest websites. The name Apache does not refer to the tribe, but is derived of the term 'a patchy server'. The default build of Apache offers just very basic functions. However the server can be expanded by a lot of modules. Therefore this can result in a patchy webserver.

0.2 Default Configuration

The webserver can be easily installed by 'apt-get install apache2'.

The server can be used for multiple sites.

- Available sites are stored in: `/etc/apache2/sites-available/`
- Enabled sites are stored in: `/etc/apache2/sites-enabled/`
- Available sites can be enabled by: `a2ensite <site-name>`
- Enabled sites can be disabled by: `a2dissite <site-name>`

In the default configuration Apache has just two sites. The sites can be accessed by the address `http://localhost` or `https://localhost` in a web-browser. At the beginning these pages just show a default-message, that the webserver is running. The contents of these sites are stored in `/var/www/`. If entering the address '`http://localhost`' the html-code of `/var/www/index.html` is loaded.

The server can be controlled by
'`/etc/init.d/apache2 { start|stop|graceful-stop|restart|reload|force-reload|start-htcacheclean|stop-htcacheclean|status}`'

- `start`: starts the sever
- `stop`: stops the server, a kill signal is sent to all subprocesses and the mainprocess.
- `graceful-stop`: a kill signal is sent to all subprocesses, to stop after finishing their actual task. New subprocesses are started with the newly loaded configuration of the apache

- restart: the server is stopped and afterwards started again.
- reload: the server reloads its configuration files
- status: shows if the server is running and which process id the main-process has

As already mentioned the function of Apache can be extended by modules.

- Available modules are stored in: `/etc/apache2/mods-available/`
- Enabled modules are stored in: `/etc/apache2/mods-enabled/`
- Available modules can be enabled by: `a2enmod <module>`
- Enabled modules can be disabled by: `a2dismod <module>`
- Some of the installable modules can be shown by:
`aptitude search apache2-mod-`

0.3 Virtual Host

A Apache server can be used to run several sites on the same machine. Apache realizes this by virtual hosts. A virtual host can be defined for an ip-address or a name. An ip-based virtual host runs each website with its own ip. A name-based virtual host runs under the same ip-address several names.

In this challenge we want to create a virtual host, which can be accessed by the url `http://<uid>.course` or simply `http://<uid>`.

In order to realize this we create the new site `/etc/apache2/sites-available/<uid>`, which has the following definition:

```
<VirtualHost * >
ServerName <uid>
ServerAlias <uid>.course

DocumentRoot /var/www

ServerAdmin <e-mail-address>
ErrorLog /var/log/apache2/error_<uid>.log
LogLevel warn

<Location>
```

Options Indexes FollowSymLinks MultiViews

Allow from all

</Location>

</VirtualHost>

Explanation of the directives:

- ServerName specifies the hostname
- ServerAlias specifies alternate hostnames
- DocumentRoot specifies the directory from which httpd will serve files. The server appends the path from the requested URL to the document root to make the path to the document.
- ServerAdmin specifies the e-mail address shown to users in the case of an error.
- ErrorLog specifies the file to which the server will log any errors it encounters.
- LogLevel specifies the priority, which is needed by messages to be recorded in the ErrorLog.
- Location is used to define the access control by url.

We want this to be our only site and therefore we can disable the sites default and default-ssl.

a2dissite default

a2dissite default-ssl

Afterwards we just need to enable the site and reload the server configuration.

a2ensite <uid>

/etc/init.d/apache2 restart

0.4 HyperText Transfer Protocol Secure

HTTPS is the abbreviation of HyperText Transfer Protocol Secure. Without an encryption protocol like HTTPS the communication between a user and a webserver is transmitted in plain text and virtually everybody in the world wide web can simply read the transmissions. Therefore it is useful to enable HTTPS for a webserver, which is the next part of the challenge. The encryption itself is realized by Transport Layer Security (TLS)/Secure

Sockets Layer (SSL). One of the advantages of HTTPS is, that no additional software is required. The default port for a https request is 443, which will be needed later.

0.4.1 Certificates

First we need to create a certificate for our webserver, which is needed for the authentication of the server by the client.

```
mkdir /etc/apache2/ssl
cd /etc/apache2/ssl
openssl req -new -x509 -nodes -out <uid>.course.crt -keyout <uid>.course.key
```

In the interactive part of this command we have to define several properties of our server:

```
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Bavaria
Locality Name (eg, city) []:Garching
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Rostlab
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:<uid>
Email Address []:<your e-mail-address>
```

0.4.2 Configuring Apache to use HTTPS

For the site, which uses ssl we have to adjust the definition:

```
<VirtualHost [youripadress]:443>
...
SSLEngine On
SSLCipherSuite HIGH:MEDIUM
SSLCertificateFile /etc/apache2/ssl/<uid>.course.crt
SSLCertificateKeyFile /etc/apache2/ssl/<uid>.course.key
...
</VirtualHost>
```

By this Apache listens on the specified ip-address and port - the default port for https - for requests to our site. Explanation of the new directives:

- SSLEngine defines if the ssl/tls Protocol Engine should be used for this site
- SSLCipherSuite specifies the Cipher Suite the client is permitted to negotiate in the SSL handshake phase
- SSLCertificateFile points to the PEM-encoded Certificate file.

- SSLCertificateKeyFile points to the PEM-encoded Private Key file for the server.

Users who try to enter our server with the http-protocol, won't find it now. Therefore a redirection from the http- to the https-version would be nice. This can be realized by adding the definition of a second virtual host after the definition of the https virtual host to the site definition. This new virtual host is a simple redirection:

```
<VirtualHost [youripaddress]:80>
ServerName <uid>.course
<Location />
Redirect permanent / https://<uid>.course/
</Location>
</VirtualHost>
```

The default installation of Apache contains the module ssl, but it is not enabled: `a2enmod ssl`

Afterwards the server needs to reload its configuration:
`/etc/init.d/apache2 restart`

0.5 PHP

PHP is a scripting language used for the dynamic creation of webpages. In order to enable this language for our server we just need to install the package. The configuration, the enabling of the module and the necessary server restart is done by aptitude: `apt-get install libapache2-mod-php5`

The next part of the challenge is to create a php-driven info-page. In order to realize this we just have to create the file `/var/www/info.php` with the content: `<?php phpinfo(); ?>`

This site is now accessible at `http://<uid>/info.php`

0.6 Access Restriction

Our site is accessible by everyone. There are several methods to limit the allowed users for our site. In principle these restrictions can be defined in every Location, Directory and File directive.

0.6.1 Restriction by IP

In order to restrict users by their ip-address you can use allow and deny. Defining single allowed ip-addresses: Allow from 192.168.16.11 192.168.16.12

Defining a whole allowed subnet: Allow from 192.168.16
 Defining a subnet by netmask and ip-address: 192.168.16.1/255.255.255.224
 This would allow the addresses from 192.168.16.1 to 192.168.16.30. There is the debian package 'ipcalc' in order to calculate the results of a netmask. The netmask defines the bits of the ip-address, which shall be used for routing. Translating 255.255.255.224 to bits results in 11111111.11111111.11111111.11100000. Therefore only the first 27 bits are used for routing and the last 5 bits are used for the hosts of the subnet. This results in a subnet of 30 usable hosts. In principle all these ways of allowing ip-addresses are also allowed to restrict ip-addresses. With 'Order Deny,Allow' one can specify which comes first.

To define the allowed ip-addresses from 192.168.16.2 to 192.168.16.19 without 192.168.16.1 and 192.168.16.10 the definition of the Location in our site could look like this:

```
<Location />
Options Indexes FollowSymLinks MultiViews
Order Allow,Deny
Allow from 192.168.16.1/255.255.255.240
Allow from 192.168.16.15 192.168.16.16 192.168.16.17 192.168.16.18 192.168.16.19
Deny from 192.168.16.1 192.168.16.10
</Location>
```

0.6.2 Restriction by User/Password

In order to restrict users, we can define userid/password pairs. Only users, who know one of these pairs are allowed to access the restricted content.

In this case we need to prepare a file with the allowed users and their passwords. With the tool htpasswd we can manage such files in quite nice way.

- Creating of the file htpasswd and adding a user with its password:
htpasswd -c /etc/apache2/htpasswd MyUserName
- Adding/updating a user/password pair to/of the existing htpasswd:
htpasswd /etc/apache2/htpasswd MyUserName2
- Deleting a user with its password from htpasswd:
htpasswd -D /etc/apache2/htpasswd MyUserName

The definition of the Location in our site could look like this:

```
<Location />
Options Indexes FollowSymLinks MultiViews
AuthType Basic
```



```
AuthName "Internal area - Authorized users only"
AuthUserFile /etc/apache2/htpasswd
Require valid-user
</Location>
```

Some explanation of the used new directives:

- AuthType defines the kind of user authentication. Basic means that the password/user is sent unencrypted to the server.
- AuthName sets the Realm to be used in the authentication. The client displays this information in the password dialog box.
- AuthUserFile points to the used password/user file.
- Require valid-user specifies, that access is only granted to users with a right password/user pair.

0.7 LDAP-Administration

In the last challenges we set up a LDAP server. This time we want to configure our webserver to provide some web-interface tools to handle the configuration of our LDAP server.

0.7.1 phpldapadmin

The first tool is phpldapadmin. It can be installed by aptitude:

```
apt-get install phpldapadmin
```

All the configuration is done automatically by Aptitude at least for a local LDAP-server.

The necessary configuration of the server for the tool is added to the server by Aptitude: /etc/apache2/conf.d/phpldapadmin Configuration files in this directory are automatically included in Apache's main configuration file /etc/apache2/apache2.conf. The tool can be accessed by the url <http://<uid>.course/phpldapadmin>

0.7.2 ldap-account-manager

The next tool is the ldap-account-manager or LAM.

This tool can be installed by aptitude:

```
apt-get install ldap-account-manager
```

Like phpldapadmin this tool also automatically configured by Aptitude and added to the server. LAM is now callable by the <http://<uid>/lam>. There are several properties in the default configuration, which should be changed or must be changed in order to run the tool with our local LDAP-server.

The lam-master-password should be changed by the following steps:
http://<uid>/lam > 'LAM configuration' > 'Edit general settings' > default password 'lam' > change the master-password ; save

The account-password and LDAP-server must be changed for our ldap-server: 'LAM configuration' > 'Edit server profiles' > default password 'lam' > 'General Settings' >

- In the section Server Settings:
 - Change Tree suffix to 'dc=course'
- In the section Security Settings:
 - Change List of valid users to 'cn=admin,dc=course'
 - Choose a new password

> save

The used nodes must be changed by the following steps in order to show the right/existing content for our LDAP-server: http://<uid>/lam > 'LAM configuration' > 'Edit server profiles' > use your new personal password > 'General Settings' >

- In the section 'Account types':
 - Delete all accounts despite Users and Groups
 - For Users change the LDAP suffix to: ou=people,dc=course
 - For Groups change the LDAP suffix to: ou=group,dc=course

> save

0.8 References

The Apache documentation:

<http://httpd.apache.org/docs/trunk/en/>

The LDAP Account Manager manual:

<http://www.ldap-account-manager.org/static/doc/manual/index.html>